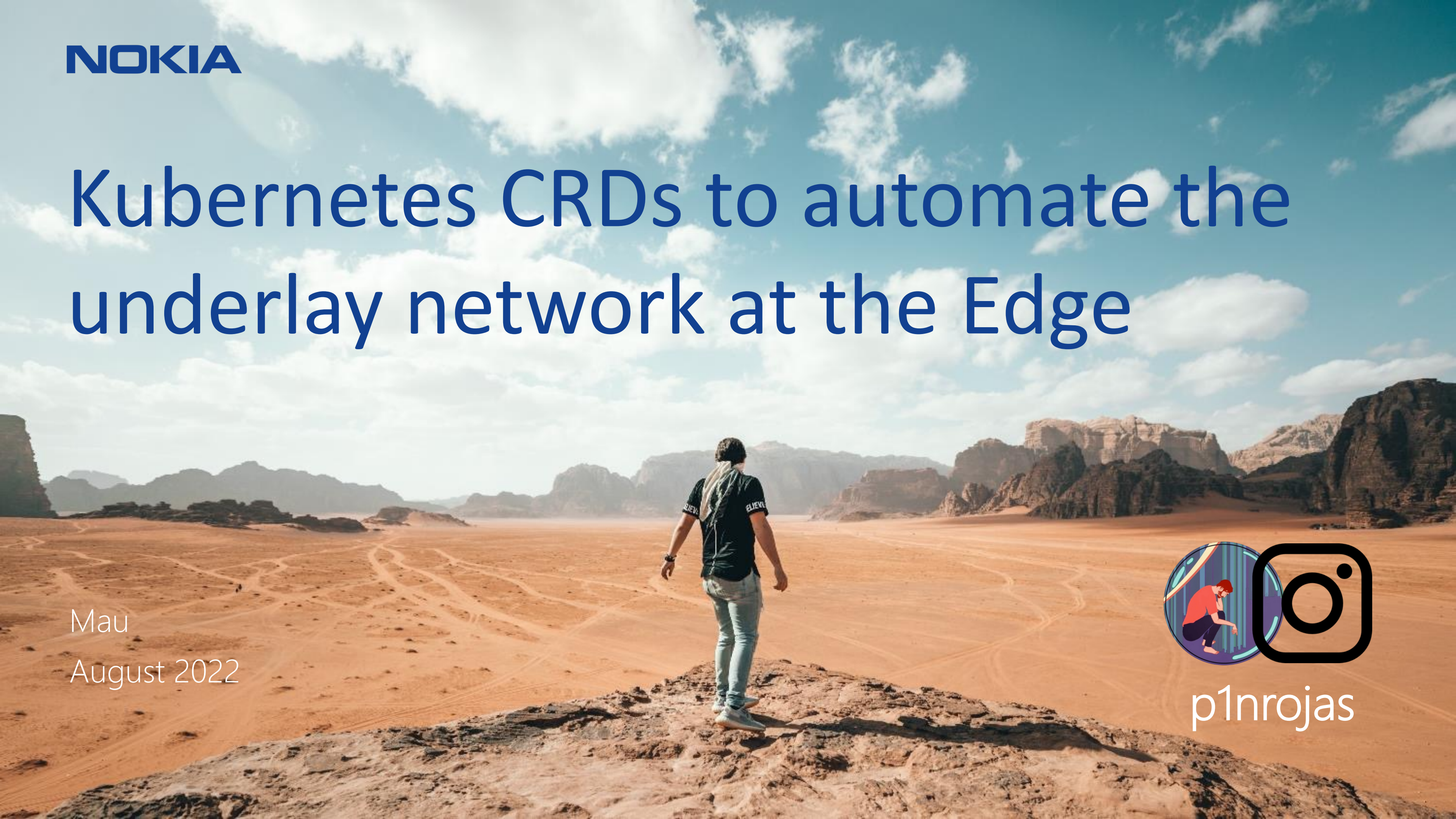
NOKIA

Kubernetes CRDs to automate the underlay network at the Edge



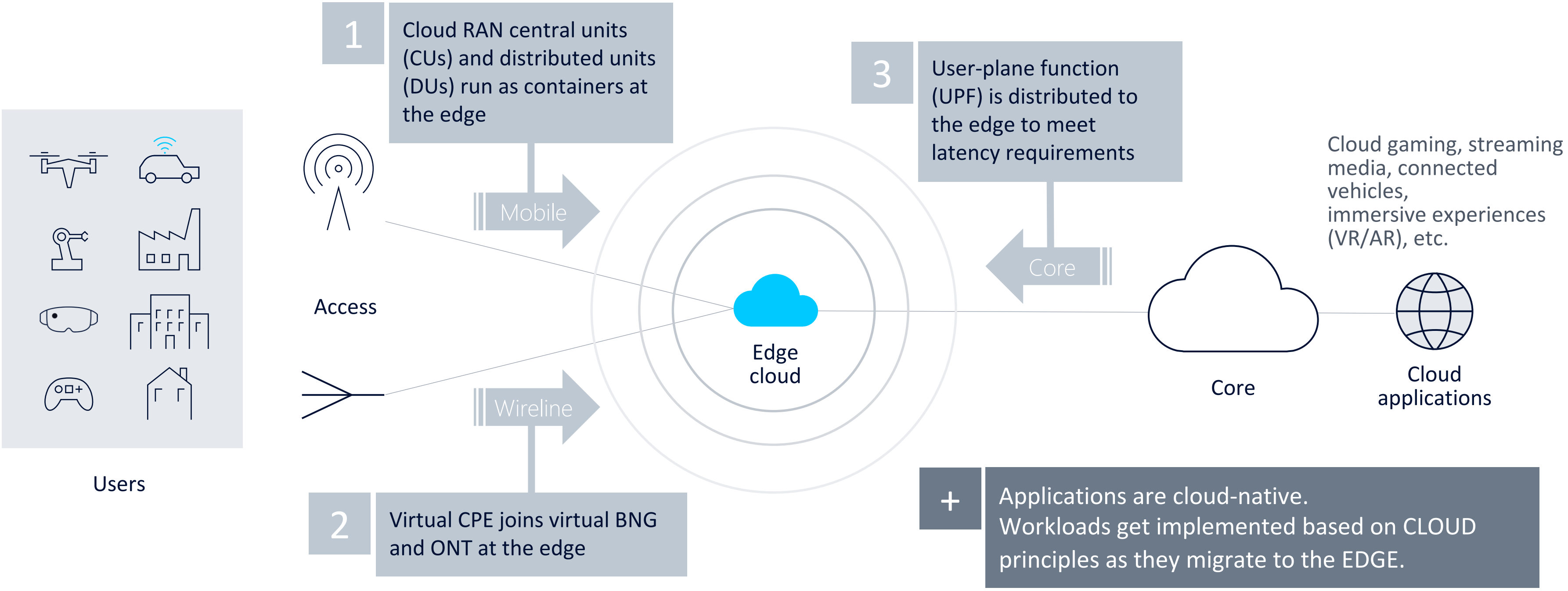
Mau

August 2022

p1nrojas

Let’s meet at the edge cloud

The edge cloud becomes a very critical piece of the infrastructure equation



1 Cloud RAN central units (CUs) and distributed units (DUs) run as containers at the edge

2 Virtual CPE joins virtual BNG and ONT at the edge

3 User-plane function (UPF) is distributed to the edge to meet latency requirements

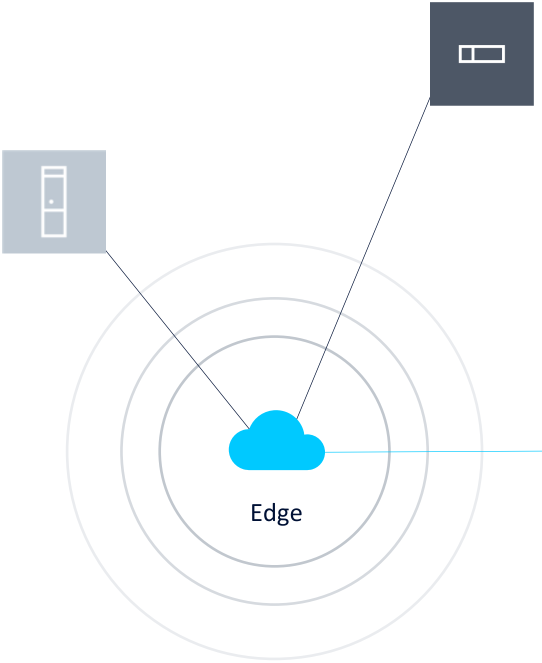
Cloud gaming, streaming media, connected vehicles, immersive experiences (VR/AR), etc.

+ Applications are cloud-native.
Workloads get implemented based on CLOUD principles as they migrate to the EDGE.

What makes this edge cloud so special

Compute and storage

- The edge is a local compute environment that builds on a **cloud-native** architecture (containers)
- Cloud management systems allow applications to **consume** workloads (compute & storage) resource on-demand
- **Kubernetes** is the most popular cloud management platform with 77% market share and growing



Networking

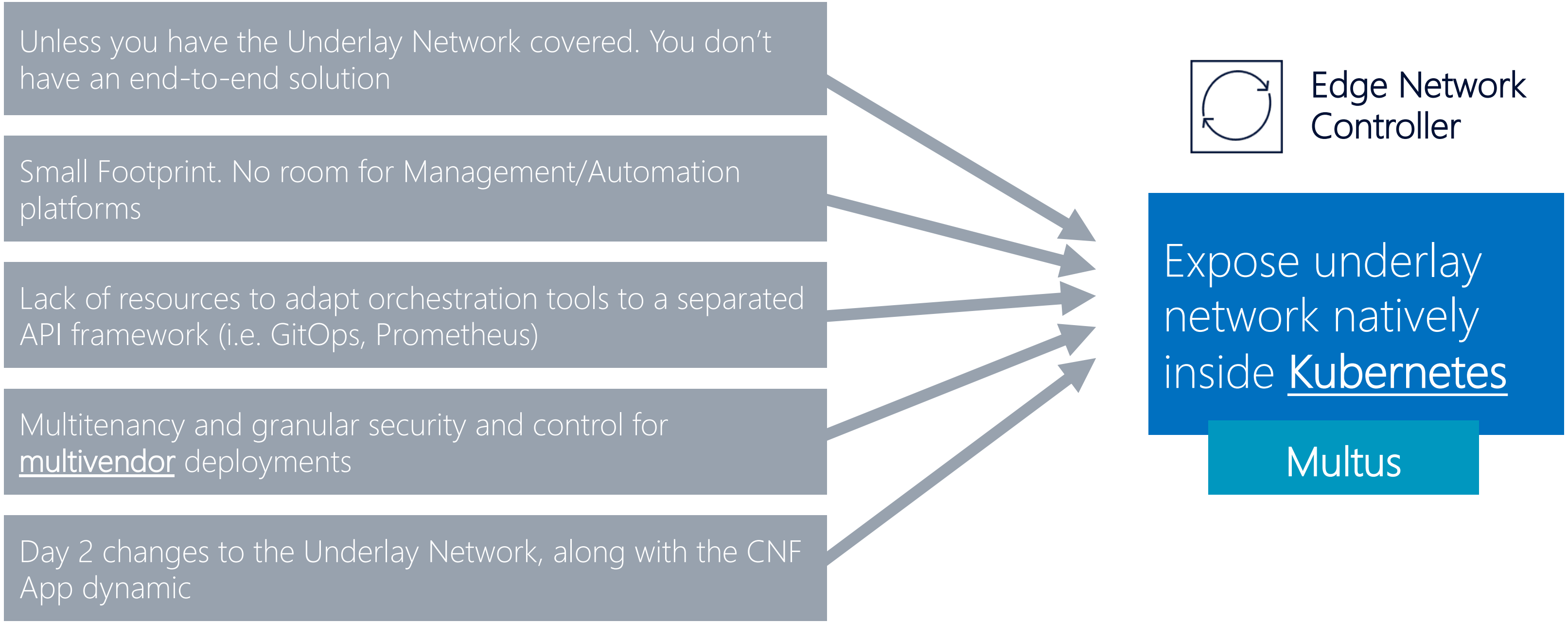
- Connect the servers hosting the workloads in the edge **and** connect to other edges and data centers

Key edge constraints & requirements

- Agility - Connections should be established automatically with compute and storage
- Efficiency - The edge is a space- and cost-constrained environment
- Self-contained - The edge should continue to run if the connection to the other data centers is lost
- Performance - Apps have stringent requirements in terms of latency and reliability

Telco CNF Apps at the Edge

Main requirements for CNF Apps at the Edge



Unless you have the Underlay Network covered. You don’t have an end-to-end solution

Small Footprint. No room for Management/Automation platforms

Lack of resources to adapt orchestration tools to a separated API framework (i.e. GitOps, Prometheus)

Multitenancy and granular security and control for multivendor deployments

Day 2 changes to the Underlay Network, along with the CNF App dynamic

Edge Network Controller

Expose underlay network natively inside Kubernetes

Multus

Kubernetes custom resource definition (CRD)

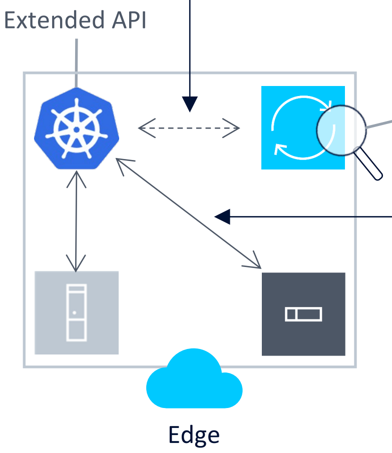
- A powerful feature introduced in Kubernetes 1.7.
- Introduce unique objects or types to meet their custom requirements

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
name: myplatforms.contoso.com
spec:
 scope: Namespaced
 versions:
   - name: v1alpha1
version.
     storage: true
     schema:
       openAPIV3Schema:
         type: object
         properties:
```

Kubernetes potential

From container orchestration to network control

Kubernetes API can be extended with custom resource and customer controllers that can be used to encode **domain knowledge (=network)** for specific applications



Operator

= client of the Kubernetes API that acts as controller for a custom resource

Custom resource

Extends Kubernetes API with network knowledge

Custom controller

Tracks desired state of the resource and tries to keep the current state of Kubernetes objects in sync with
the desired state

Kubernetes API can be used to **configure the network** and enforce policies (security rules, traffic policy and traffic engineering rules, service chaining rules)

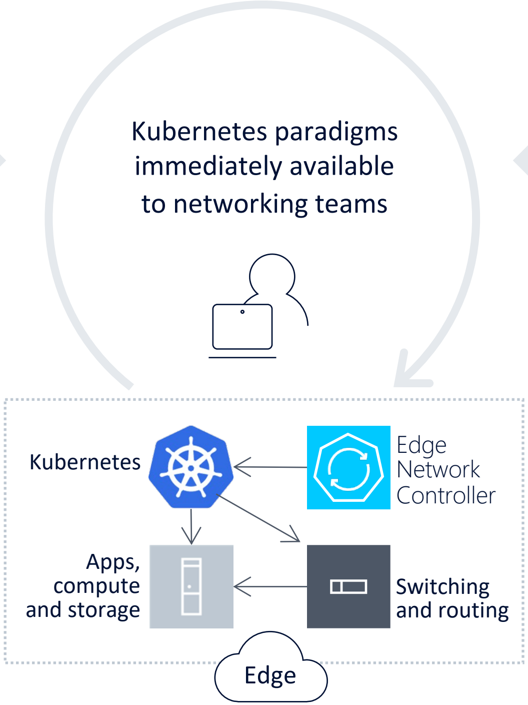
Leveraging the Kubernetes ecosystem

GitOps collaborative across involved teams

gitops

Enables organizations to continuously deliver software applications while efficiently managing IT and network infrastructure

- Declarative
- Versioned and immutable
- Pulled automatically
- Continuously reconciled



Kubernetes paradigms immediately available to networking teams

Cloud Native Computing Foundation

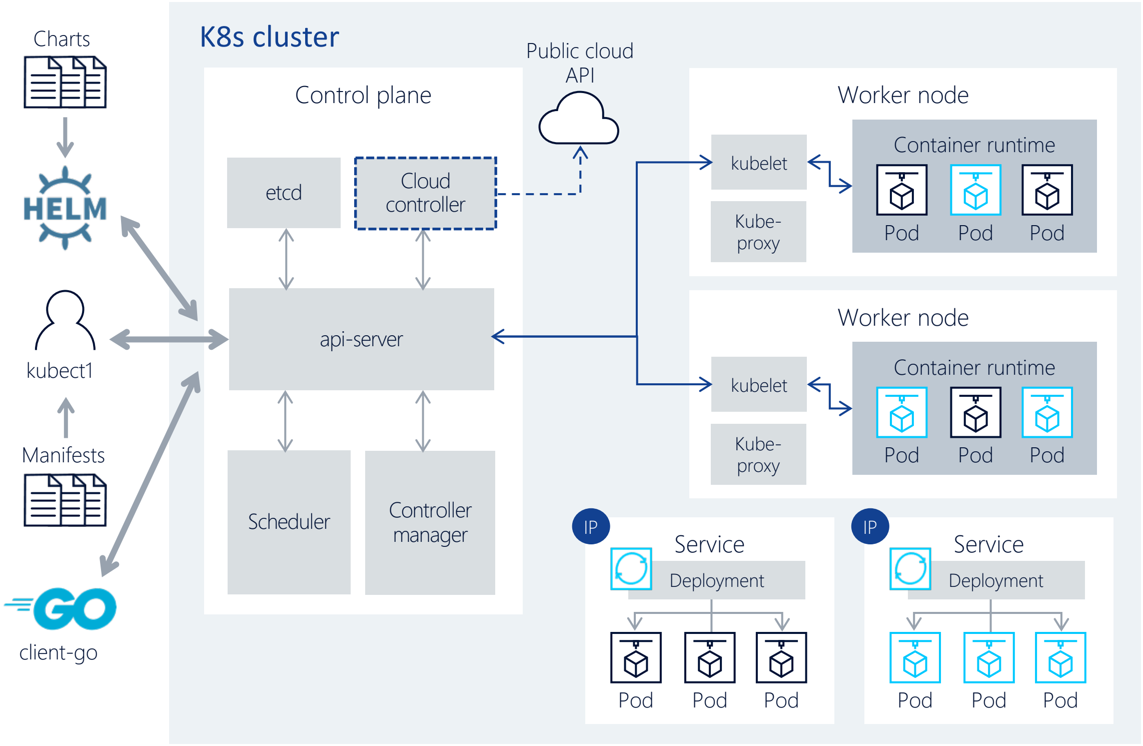
Prometheus

Fluentd

Elastic

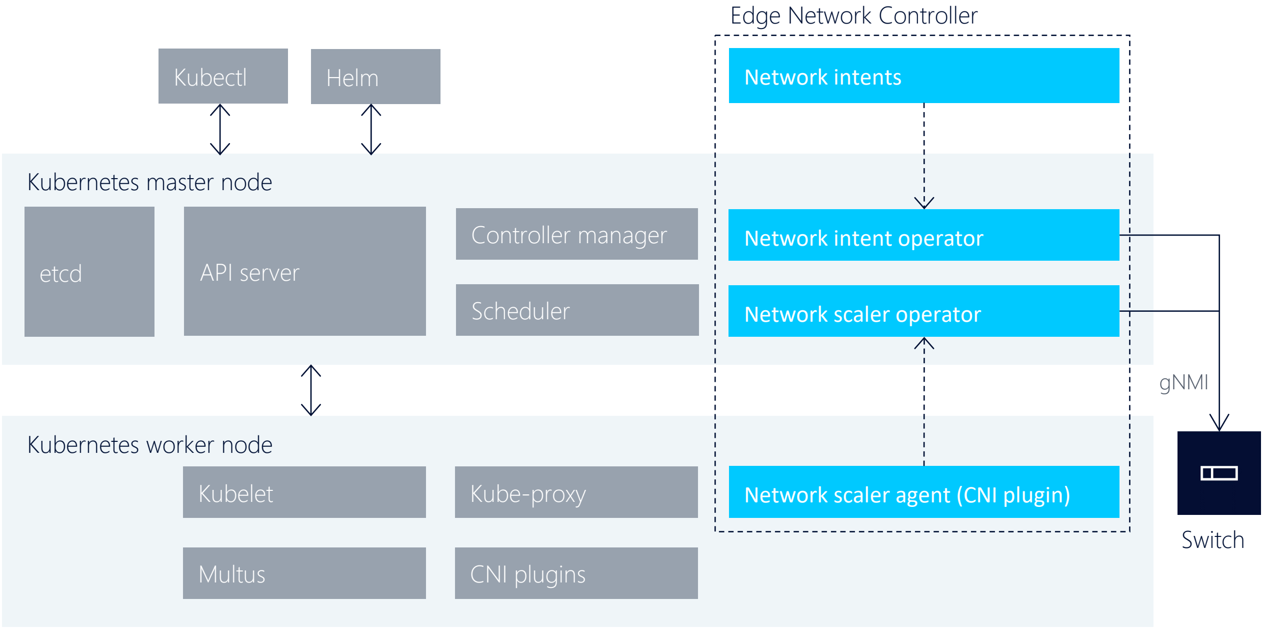
Monitoring, logging and assurance through CNCF-based industry proven tools

Kubernetes architecture



Architecture of the Edge Network Controller

Extending Kubernetes to enable full network control

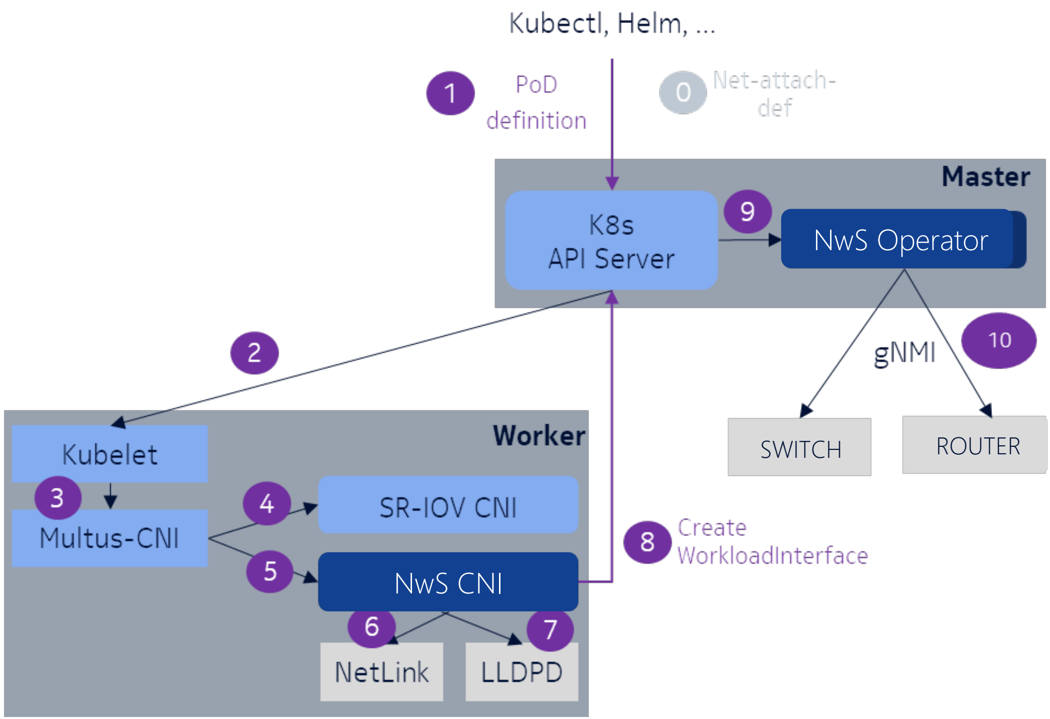


An **operator** is a client of the Kubernetes API that acts as controller for a custom resource

- **Network intent operator** allows exposure of the YANG tree of the switch and its configuration using Kubernetes API paradigm
- **Network scaler** (operator + agent) is a lightweight application designed to react to events and configure the switch appropriately

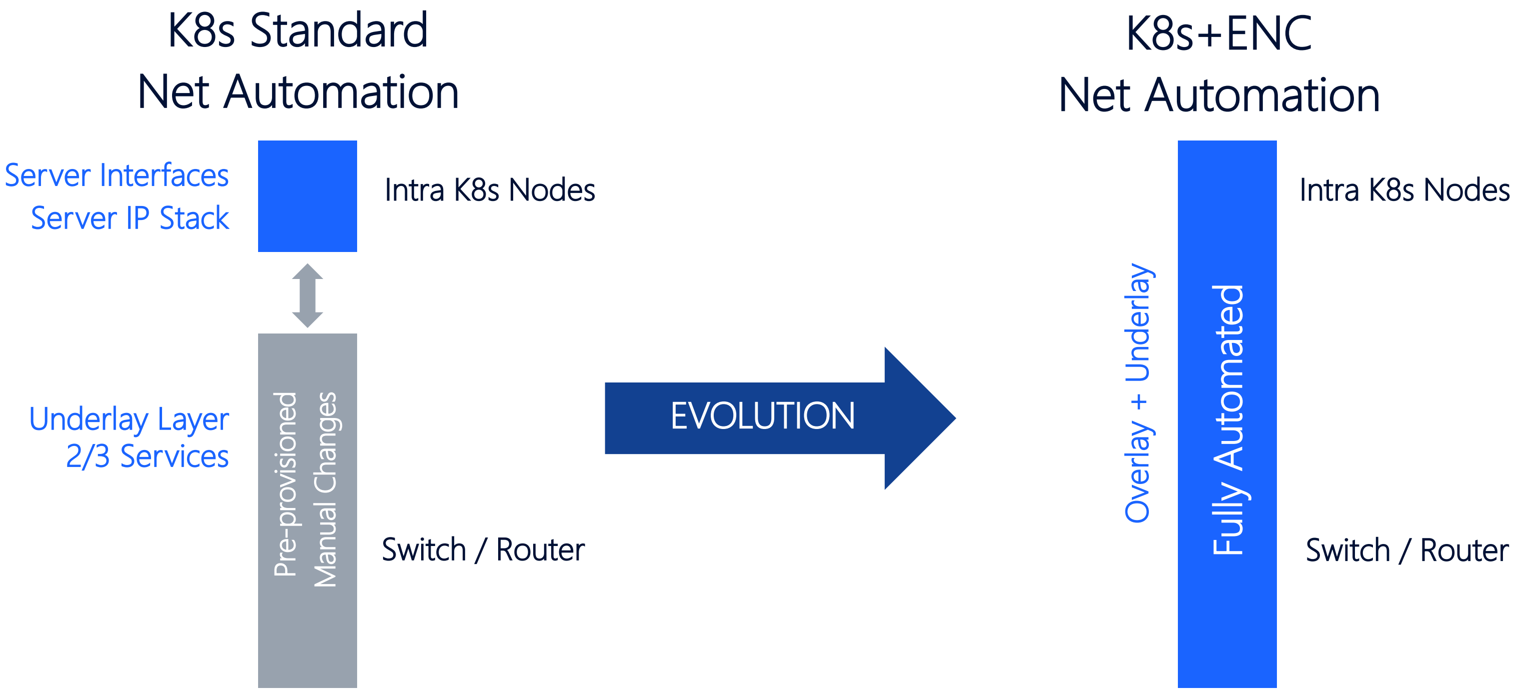
Scaler: How does it work? (2)

Dynamic configuration flow



1. Deploy pod
2. Pod is scheduled to worker node
3. Kubelet requests Multus to set up networking
4. Multus calls first CNI as defined in the configuration list
5. Multus calls NwS CNI as next CNI
6. NwS CNI resolves the physical port and vlan from the pod interface information received from Multus
7. NwS CNI retrieves the switch and port ID
8. NwS CNI creates a k8s ‘WorkloadInterface’ custom resource
9. K8S Kubernetes API triggers the NwS controller that owns the the WorkloadInterface CRD
10. The NwS controller reconciles the requested WorkloadInterface intent with switch configuration via gNMI (see further)

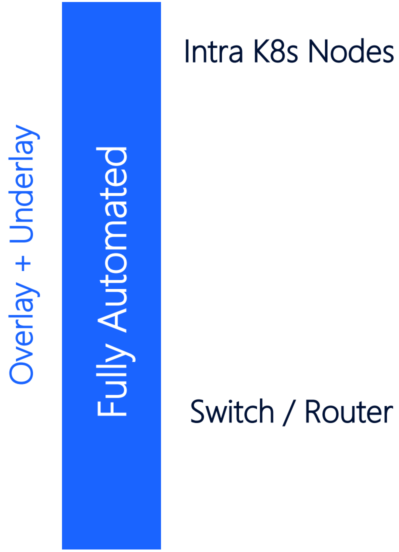
Edge Network Controller: Overview



Edge Network Controller: Overview

Today	ENC inherit Value
Limited context from application requirements	**Declarative Intent** Network Design by the minute
Setup and changes are managed by different teams	**Unify deploy & infra** One Pipeline Investment
Manual tracking is prone to errors and hard to troubleshoot	**Single source of Truth** Consistent Communication between teams

K8s+ENC
Net Automation



Value proposition of the Edge Network Controller

Role and responsibility of involved teams

Consumers of the network

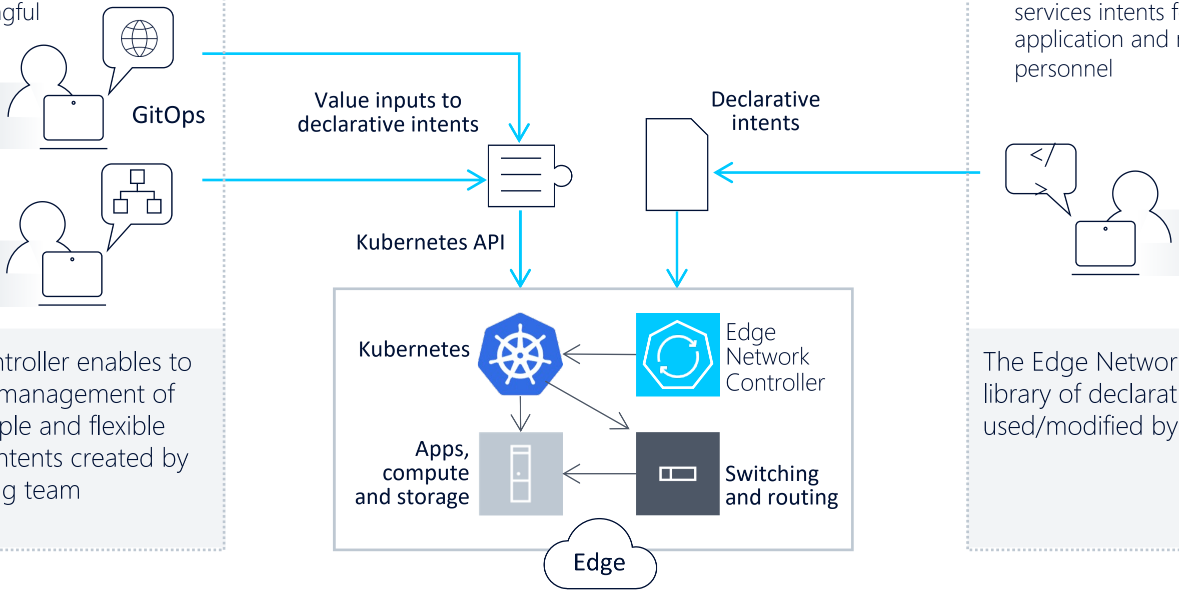
Configure the network using minimal, meaningful input parameters

IT / apps department

Network operations team

The Edge Network Controller enables to automate the lifecycle management of the IP network in a simple and flexible way using declarative intents created by the network engineering team

Delegation of authority



GitOps

Value inputs to declarative intents

Declarative intents

Kubernetes API

Kubernetes

Edge Network Controller

Apps, compute and storage

Switching and routing

Edge

Designers of the network

Offer access to a specific set of network services intents for consumption by the application and network operational personnel

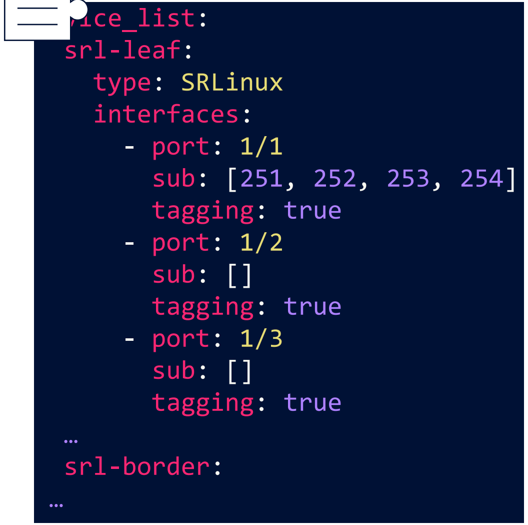
Network engineers & architects

The Edge Network Controller brings a library of declarative intents that can be used/modified by the operations team.

Automated device provisioning with ENC-NwI

Leverage declarative templating engine to generate k8s resources

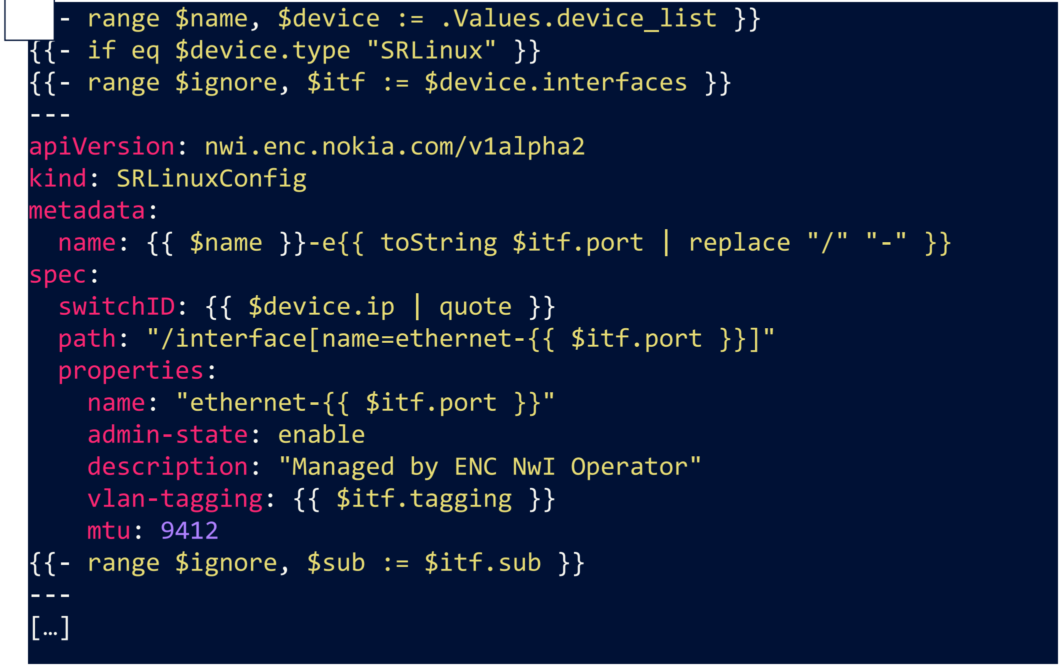
Network operations team



```
vice_list:
  srl-leaf:
    type: SRLinux
    interfaces:
      - port: 1/1
        sub: [251, 252, 253, 254]
        tagging: true
      - port: 1/2
        sub: []
        tagging: true
      - port: 1/3
        sub: []
        tagging: true
  …
  srl-border:
…
```

values.yaml

Network engineers & architects



```
- range $name, $device := .Values.device_list }}
{{- if eq $device.type "SRLinux" }}
{{- range $ignore, $itf := $device.interfaces }}
---
apiVersion: nwi.enc.nokia.com/v1alpha2
kind: SRLinuxConfig
metadata:
  name: {{ $name }}-e{{ toString $itf.port | replace "/" "-" }}
spec:
  switchID: {{ $device.ip | quote }}
  path: "/interface[name=ethernet-{{ $itf.port }}]"
  properties:
    name: "ethernet-{{ $itf.port }}"
    admin-state: enable
    description: "Managed by ENC NwI Operator"
    vlan-tagging: {{ $itf.tagging }}
    mtu: 9412
{{- range $ignore, $sub := $itf.sub }}
---
[…]
```

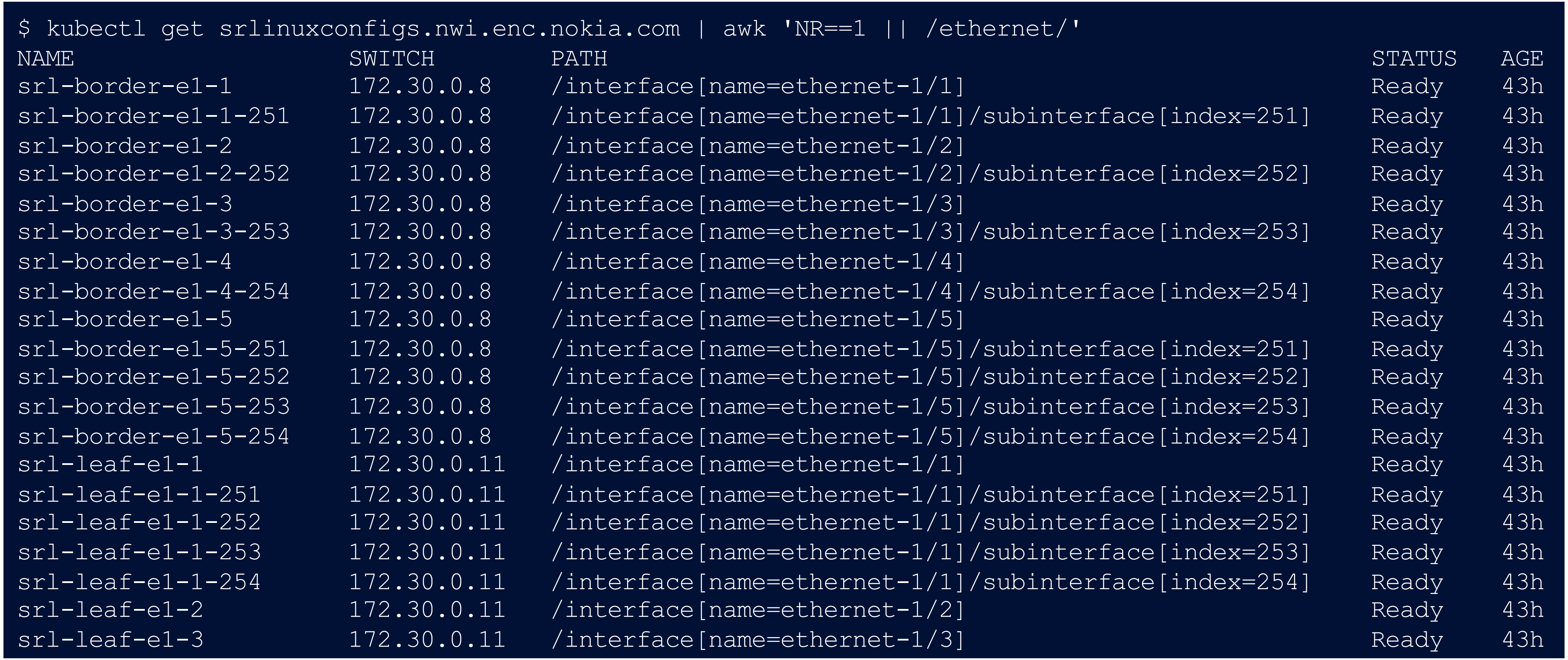
template/srl-interface-config.yaml

Network resources are managed as native k8s resources

e.g., part of automated deployments to describe application’s network SLAs

IT / apps department

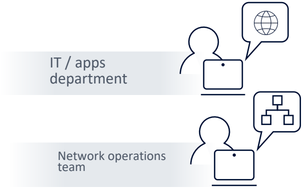
Network operations team



```
$ kubectl get srlinuxconfigs.nwi.enc.nokia.com | awk 'NR==1 || /ethernet/'
NAME                     SWITCH         PATH                                                   STATUS    AGE
srl-border-e1-1          172.30.0.8     /interface[name=ethernet-1/1]                          Ready     43h
srl-border-e1-1-251      172.30.0.8     /interface[name=ethernet-1/1]/subinterface[index=251]  Ready     43h
srl-border-e1-2          172.30.0.8     /interface[name=ethernet-1/2]                          Ready     43h
srl-border-e1-2-252      172.30.0.8     /interface[name=ethernet-1/2]/subinterface[index=252]  Ready     43h
srl-border-e1-3          172.30.0.8     /interface[name=ethernet-1/3]                          Ready     43h
srl-border-e1-3-253      172.30.0.8     /interface[name=ethernet-1/3]/subinterface[index=253]  Ready     43h
srl-border-e1-4          172.30.0.8     /interface[name=ethernet-1/4]                          Ready     43h
srl-border-e1-4-254      172.30.0.8     /interface[name=ethernet-1/4]/subinterface[index=254]  Ready     43h
srl-border-e1-5          172.30.0.8     /interface[name=ethernet-1/5]                          Ready     43h
srl-border-e1-5-251      172.30.0.8     /interface[name=ethernet-1/5]/subinterface[index=251]  Ready     43h
srl-border-e1-5-252      172.30.0.8     /interface[name=ethernet-1/5]/subinterface[index=252]  Ready     43h
srl-border-e1-5-253      172.30.0.8     /interface[name=ethernet-1/5]/subinterface[index=253]  Ready     43h
srl-border-e1-5-254      172.30.0.8     /interface[name=ethernet-1/5]/subinterface[index=254]  Ready     43h
srl-leaf-e1-1            172.30.0.11    /interface[name=ethernet-1/1]                          Ready     43h
srl-leaf-e1-1-251        172.30.0.11    /interface[name=ethernet-1/1]/subinterface[index=251]  Ready     43h
srl-leaf-e1-1-252        172.30.0.11    /interface[name=ethernet-1/1]/subinterface[index=252]  Ready     43h
srl-leaf-e1-1-253        172.30.0.11    /interface[name=ethernet-1/1]/subinterface[index=253]  Ready     43h
srl-leaf-e1-1-254        172.30.0.11    /interface[name=ethernet-1/1]/subinterface[index=254]  Ready     43h
srl-leaf-e1-2            172.30.0.11    /interface[name=ethernet-1/2]                          Ready     43h
srl-leaf-e1-3            172.30.0.11    /interface[name=ethernet-1/3]                          Ready     43h
```

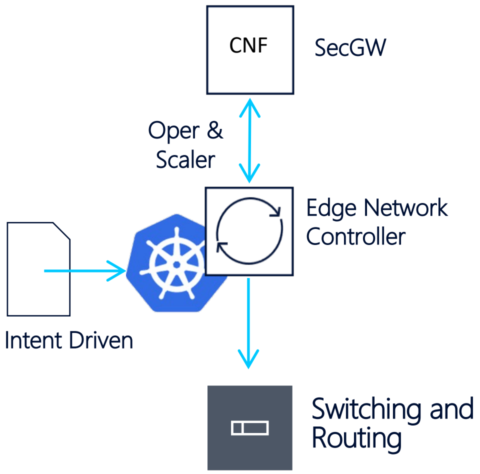
Network resources are exposed as native k8s resources

Can be consumed by applications, e.g., to track network state



```
$ kubectl get srlinuxconfigs.nwi.enc.nokia.com srl-leaf-e1-1 -o yaml
apiVersion: nwi.enc.nokia.com/v1alpha2
kind: SRLinuxConfig
metadata:
[…]
  name: srl-leaf-e1-1
  namespace: default
spec:
  path: /interface[name=ethernet-1/1]
  properties:
    admin-state: enable
    mtu: 9412
    name: ethernet-1/1
    vlan-tagging: true
  switchID: 172.30.0.11
status:
  conditions:
  - lastTransitionTime: "2022-03-14T14:34:41Z"
    message: ""
    reason: Created
    status: "True"
    type: Ready
```

Some Use Cases


CNF

SecGW

Oper & Scaler

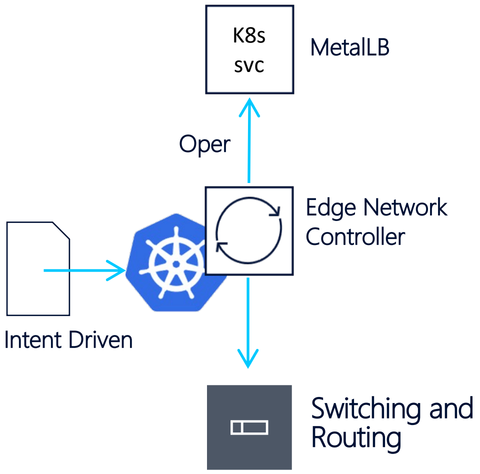
Edge Network Controller

Intent Driven

Switching and Routing

Home Enterprise Network

Telcos / HealthCare


K8s svc

MetalLB

Oper

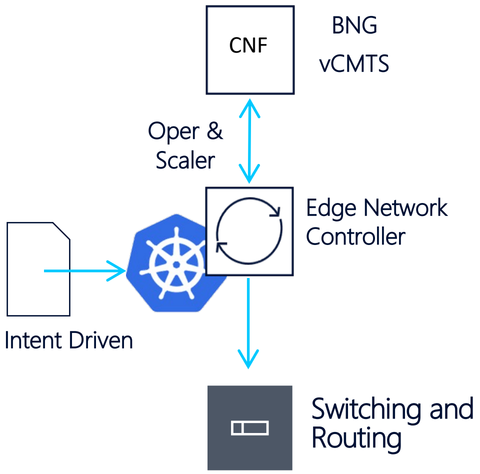
Edge Network Controller

Intent Driven

Switching and Routing

Load Balancer BGP mode

Enterprise


CNF

BNG

vCMTS

Oper & Scaler

Edge Network Controller

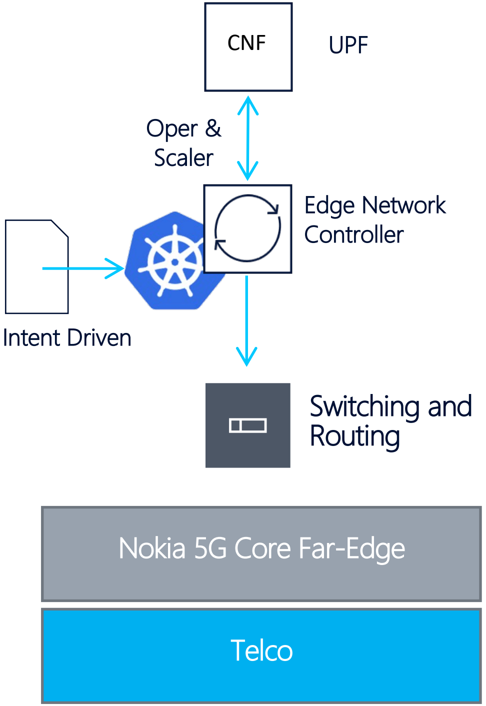
Intent Driven

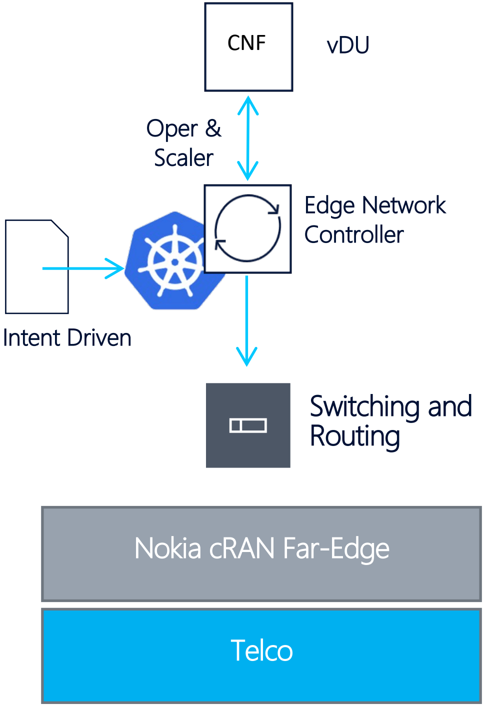
Switching and Routing

High Speed Data

Telcos (MSOs)

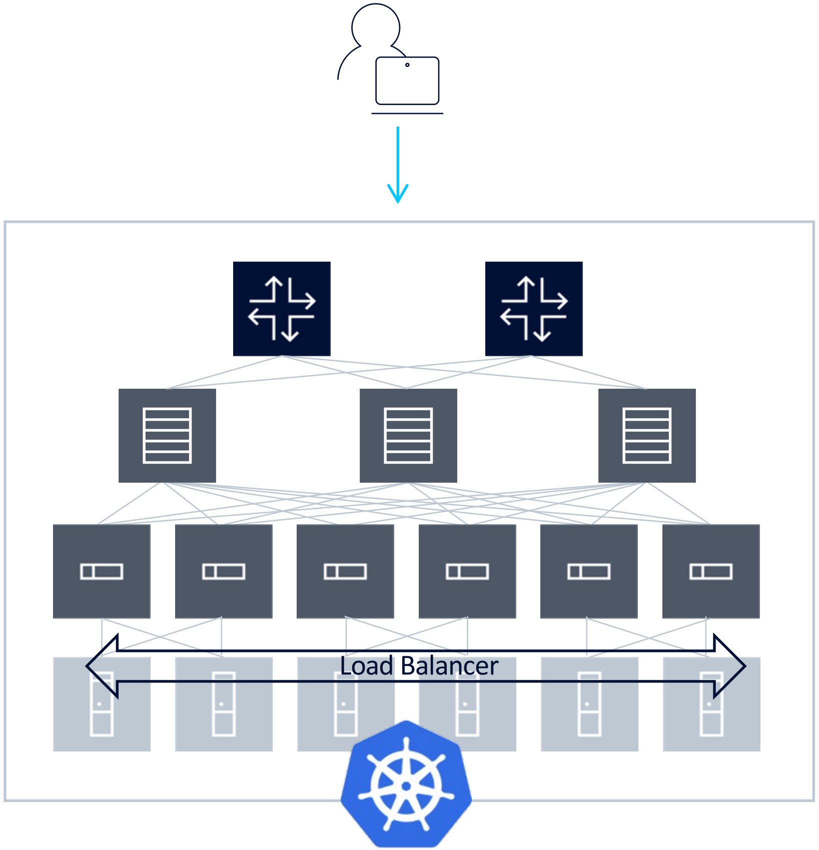
Some Use Cases (cont.)





BGP Mode K8s Load Balancer

- BGP-based leafs implement stateless load balancing
  - Add Ingress for Stateful
- No Bottlenecks.
  - iBGP brings distribution across the Network Fabric.
- Resilient
  - Fast failover
  - BFD support (Experimental and no included in this demo)
- Enables True Load Balancing via ECMP
- Traffic control: Cluster vs Local



Workforces split between offices and homes are the new norm

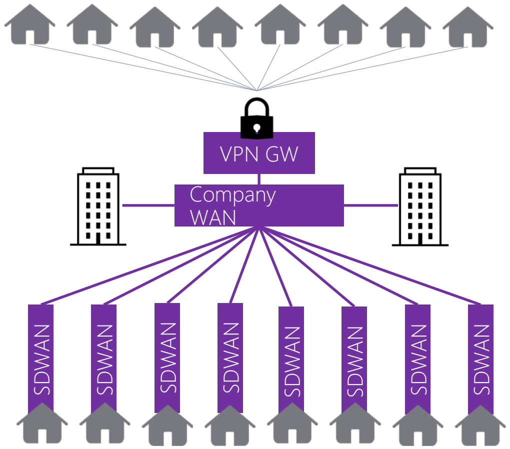
The COVID19 pandemic only acted as accelerant

Home are also becoming “micro branch offices”

- Secure access to company resources is required
- Company traffic must coexist with private use

E2E VPNs are challenging

- Expensive, e.g., licenses, support costs
- Complex scalability, e.g., on-premises HW appliances
- Operational complexity, e.g., keys/certificates, active troubleshooting of network-induced issues such as CGNAT/PMTU



SD-WAN gateways in the home are not a solution either…

Key takeaways

The Edge Network Controller has unique characteristics inherited from Kubernetes to help CSPs automate their edge cloud networks

Lightweight.

Minimal resources on a server

- An application of Kubernetes, hosted on the same cluster
- Strong requirement for edge locations with very limited space

Powerful.

As powerful as Kubernetes can be

- Leverage its declarative intent-based approach
- Benefits from its entire ecosystem and tooling

Nimble.

Tied to the applications it supports

- Autonomous event-driven network automation
- Storage, compute and network in the same lifecycle management

Thanks

p1nrojas